CLAIMS

What is claimed is:

1. A method of transferring data from a markup language file having a hierarchical structure to a relational database, said hierarchical structure comprising a tree or forest of nodes on which depth first search imposes a total ordering, with some nodes designated as repeating nodes, and said method comprising:

partitioning said hierarchical structure into sections, wherein each section is dedicated to at least one leaf node of said hierarchical structure, and wherein two non-repeating leaf nodes that are adjacent in frontier order and have the same parent are contained in the same section, frontier order being the order in which leaf nodes are encountered in a depth first search of said hierarchical structure;

allocating a memory section for each of said sections of said hierarchical structure according to the data types of the nodes in the section;

after completing said partitioning and allocating, parsing said markup language file to produce a stream of data pairs, wherein each of said data pairs comprises an element of node data and an element of node location information, and wherein said node location information indicates the location of the corresponding node within said hierarchical structure;

while performing said parsing process, loading said node data into the memory section allocated for the section containing the corresponding node location as said data pairs are output from said parsing process; and

transferring said node data from said sections to said relational database, wherein information is transferred from one section as soon as said loading process completes loading at least one element of node data to said one memory section and an end of section indicator has been encountered by said parsing process.

2. The method in claim 1, wherein said partitioning said hierarchical structure into sections, wherein each section is dedicated to at least one leaf node of said hierarchical structure, and wherein two non-repeating leaf nodes that are adjacent in frontier order and have the same parent

are contained in the same section, frontier order being the order in which leaf nodes are encountered in a depth first search of said hierarchical structure.

3. The method in claim 1, further comprising erasing said memory section, wherein a first memory section is erased only when an end of section indicator has been encountered by said parsing process, a new corresponding data pair is produced by said parsing process, and the node data of said data pair is ready to be loaded in said first memory section.

4. The method in claim 1, wherein said transferring said node data from said sections to said relational database, wherein information is transferred from one section as soon as said loading process completes loading at least one element of node data to said one memory section and an end of section indicator has been encountered by said parsing process, wherein an end of section indicator is encountered when the parsing process produces either a node location from a different section or a node location at or preceding the last of the at least one node location in the one section in depth first search order.

5. The method in claim 1, wherein said node location information of said data pairs comprises leaf nodes of said hierarchical data structure.

6. The method in claim 1, wherein in said partitioning process any two non-repeating leaf nodes of said hierarchical structure that are adjacent in frontier order and have the same repeating ancestors are in the same section.

7. The method in claim 1, wherein said parsing process relocates all data in said hierarchical structure to the leaf nodes of said hierarchical structure.

8. A method of transferring data from a markup language file having a hierarchical structure to a relational database, said hierarchical structure comprising a tree or forest of nodes on which depth first search imposes a total ordering, with some nodes designated as repeating nodes, and said method comprising:

partitioning said hierarchical structure into sections, wherein each section is dedicated to at least one leaf node of said hierarchical structure, and wherein two non-repeating leaf nodes that are adjacent in frontier order and have the same parent are contained in the same section, frontier order being the order in which leaf nodes are encountered in a depth first search of said hierarchical structure;

allocating a memory section for each said section of said hierarchical structure according to the data types of the nodes in the section;

after completing said partitioning and allocating, parsing said markup language file to produce a stream of data pairs, wherein each of said data pairs comprises an element of node data and an element of node location information, and wherein said node location information indicates the location of the corresponding node within said hierarchical structure;

loading said node data into corresponding sections as said node data elements are output from said parsing process; and

transferring said node data from said sections to said relational database, wherein information is transferred from one section as soon as said loading process completes loading at least one element of node data to said one memory section and an end of section indicator has been encountered by said parsing process.

9. The method in claim 8, wherein said partitioning said hierarchical structure into sections, wherein each section is dedicated to at least one leaf node of said hierarchical structure, and wherein two non-repeating leaf nodes that are adjacent in frontier order and have the same parent are contained in the same section, frontier order being the order in which leaf nodes are encountered in a depth first search of said hierarchical structure.

10. The method in claim 8, further comprising erasing said memory section, wherein a first memory section is erased only when an end of section indicator has been encountered by said parsing process, a new corresponding data pair is produced by said parsing process, and the node data of said data pair is ready to be loaded in said first memory section.

11. The method in claim 8, wherein said transferring said node data from said sections to said relational database, wherein information is transferred from one section as soon as said loading process completes loading at least one element of node data to said one memory section and an end of section indicator has been encountered by said parsing process, wherein an end of section indicator is encountered when the parsing process produces either a node location from a different section or a node location at or preceding the last of the at least one node location in the one section in depth first search order.

12. The method in claim 8, wherein said node location information of said data pairs comprises leaf nodes of said hierarchical data structure.

13. The method in claim 8, wherein in said partitioning process any two non-repeating leaf nodes of said hierarchical structure that are adjacent in frontier order and have the same repeating ancestors are in the same section.

14. The method in claim 8, wherein said parsing process relocates all data in said hierarchical structure to the leaf nodes of said hierarchical structure.

15. A method of transferring data from a markup language file having a hierarchical structure to a relational database, said hierarchical structure comprising a tree or forest of nodes on which depth first search imposes a total ordering, with some nodes designated as repeating nodes, and said method comprising:

partitioning said hierarchical structure into sections, wherein each section is dedicated to at least one leaf node of said hierarchical structure, and wherein two non-repeating leaf nodes that are adjacent in frontier order and have the same parent are contained in the same section, frontier order being the order in which leaf nodes are encountered in a depth first search of said hierarchical structure;

allocating a memory section for each said section of said hierarchical structure according to the data types of the nodes in the section;

after completing said partitioning and allocating, parsing said markup language file to produce a stream of data pairs, wherein each of said data pairs comprises an element of node data and an element of node location information, and wherein said node location information indicates the location of the corresponding node within said hierarchical structure;, wherein each of said data pairs is in the form (tag, field), and wherein said field represents node data and said tag represents the location of corresponding node data within said hierarchical structure;

loading said data pairs into corresponding sections as said data pairs are output from said parsing process; and

transferring said node data from said sections to said relational database, wherein information is transferred from one section as soon as said loading process completes loading at least one element of node data to said one memory section and begins loading a different element of node data to a different memory section.

16. The method in claim 15, wherein said partitioning is based on a document type definition file, separate from said hierarchical file, wherein said document type definition file comprises said hierarchical structure.

17. The method in claim 15, further comprising erasing said sections, wherein a first section is erased only when a new corresponding data pair is produced by said parsing process and is ready to be loaded in said first section.

18. The method in claim 15, wherein said transferring process is performed as soon as the loading of a corresponding data pair into a corresponding section is complete, as indicated by said end of section indicators.

19. The method in claim 15, wherein said data pairs comprise leaf nodes of said hierarchical structure.

20. The method in claim 15, wherein leaf nodes of said hierarchical structure include repeating nodes and wherein a different section is exclusively dedicated to each of said repeating nodes.

21. The method in claim 15, wherein said parsing process relocates all data in said hierarchical structure to the leaf nodes of said hierarchical structure.

22. A method of altering the hierarchical structure of a markup language file for being processed into a relational database, said method comprising:

identifying repeating nodes and non-repeating nodes within said hierarchical structure; and

reorganizing said hierarchical structure such that non-repeating nodes are positioned before repeating nodes within each hierarchal level of said hierarchical structure.

23. The method in claim 22, wherein said hierarchical structure comprises the tree structure having at least one root node, at least one branch node proceeding from said root node, and least one leaf node proceeding from said branch node.

24. The method in claim 23, wherein said process of reorganizing said hierarchical structure comprises:

reorganizing root nodes such that non-repeating root nodes are positioned before repeating root nodes;

after reorganizing said root nodes, reorganizing branch nodes such that non-repeating branch nodes are positioned before repeating branch nodes; and

after reorganizing said branch nodes, reorganizing leaf nodes such that non-repeating leaf nodes are positioned before repeating leaf nodes.

25. The method in claim 22, wherein said hierarchical structures is contained within a document type definition (DTD) file.

26. A method of transferring data from a markup language file having a hierarchical structure to a relational database said method comprising:

partitioning said hierarchical structure into sections;

allocating a memory section for each of said sections of said hierarchical structure according to the data types of the nodes in the section;

after completing said partitioning and allocating, parsing said markup language file to produce a stream of data pairs while performing said parsing process, loading said node data into the memory section allocated for the section containing the corresponding node location as said data pairs are output from said parsing process; and

transferring said node data from said sections to said relational database.

27. A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform a method of transferring data from a markup language file having a hierarchical structure to a relational database, said hierarchical structure comprising a tree or forest of nodes on which depth first search imposes a total ordering, with some nodes designated as repeating nodes, and said method comprising:

partitioning said hierarchical structure into sections, wherein each section is dedicated to at least one leaf node of said hierarchical structure, and wherein two non-repeating leaf nodes that are adjacent in frontier order and have the same parent are contained in the same section, frontier order being the order in which leaf nodes are encountered in a depth first search of said hierarchical structure;

allocating a memory section for each said section of said hierarchical structure according to the data types of the nodes in the section;

after completing said partitioning and allocating, parsing said markup language file to produce a stream of data pairs, wherein each of said data pairs comprises an element of node data and an element of node location information, and wherein said node location information indicates the location of the corresponding node within said hierarchical structure;

while performing said parsing process, loading said node data into the memory section allocated for the section containing the corresponding node location as said data pairs are output from said parsing process; and

transferring said node data from said sections to said relational database, wherein information is transferred from one section as soon as said loading process completes loading at least one element of node data to said one memory section and begins loading a different element of node data to a different memory section.

28. The program storage device in claim 27, wherein said method further comprises partitioning said hierarchical structure into sections, wherein each section is dedicated to at least one leaf node of said hierarchical structure, and wherein two non-repeating leaf nodes that are adjacent in frontier order and have the same parent are contained in the same section, frontier order being the order in which leaf nodes are encountered in a depth first search of said hierarchical structure.

29. The program storage device in claim 27, wherein said method further erasing said memory section, wherein a first memory section is erased only when an end of section indicator has been encountered by said parsing process, a new corresponding data pair is produced by said parsing process, and the node data of said data pair is ready to be loaded in said first memory section.

30. The program storage device in claim 27, wherein said method further comprises transferring said node data from said sections to said relational database, wherein information is transferred from one section as soon as said loading process completes loading at least one element of node data to said one memory section and begins loading a different element of node data to a different memory section.

31. The program storage device in claim 27, wherein said method further comprises node location information of said data pairs comprise leaf nodes of said hierarchical data structure.

32. The program storage device in claim 27, wherein said method further comprises partitioning process any two non-repeating leaf nodes of said hierarchical structure that are adjacent in frontier order and have the same repeating ancestors are in the same section.

33. The program storage device in claim 27, wherein said method further comprises parsing process relocates all data in said hierarchical structure to the leaf nodes of said hierarchical structure.